JA 999009

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 878 768 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
18.11.1998 Bulletin 1998/47

(51) Int Cl.[6]: **G06F 17/30**

(21) Application number: **98303229.3**

(22) Date of filing: **27.04.1998**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **13.05.1997 US 855244**

(71) Applicant: **INTERNATIONAL BUSINESS MACHINES CORPORATION**
Armonk, NY 10504 (US)

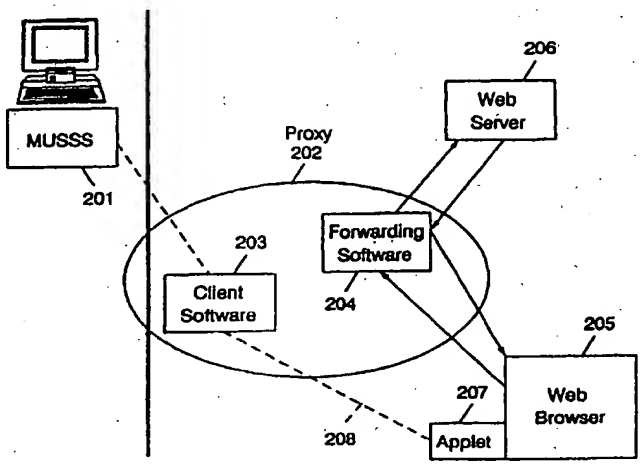
(72) Inventors:
- **Cuomo, Gennaro A.**
  Apex, North Carolina 27502 (US)
- **Nguyen, Binh Q.**
  Cary, North Carolina 27513 (US)
- **Nguyen, Hung D.**
  Cary, North Carolina 27511 (US)
- **Singhal, Sandeep K.**
  Raleigh, North Carolina 27605-1271 (US)

(74) Representative: **Zerbi, Guido Maria**
Intellectual Property Department,
IBM United Kingdom Ltd.,
Hursley Park
Winchester, Hampshire SO21 2JN (GB)

(54) Method and system for portably enabling awareness, touring, and conferencing over the world-wide web using proxies and shared-state servers

(57) Collaboration capabilities (both "push" and "pull" of information) are portably added to World-Wide Web content. In particular, a persistent connection is maintained to a shared state server for retrieving data via a "push" mechanism which maintains the freedom to access arbitrary Web content via the "pull" mechanism. Moreover, the collaboration is provided in a portable manner, in the same way that all other Web content is portable both across hardware platforms and across browser implementations.

FIG. 2



EP 0 878 768 A2

Description

Field of the Invention

This invention relates in general to computer software, and in particular to a method and system for enabling multi-user collaboration over the World-Wide Web. More particularly, the present invention relates to a method and system by which awareness, touring, and conferencing can be performed in a portable manner.

Background of the Invention

Two techniques are in use today for delivering data from a supplier to a consumer: "push" and "pull". Under a "push" model, which is used by services such as PointCast, Castanet, and TIB, information is sent to the recipient whenever it becomes available at the source. Under a "pull" model, which is used by services such as Excite, the data is only transmitted to the recipient in response to an explicit request. These two methods excel at a variety of data retrieval tasks, from supporting random searches ("pull") to delivering dynamic information updates ("push").

However, these data retrieval metaphors are insufficient to support real-time collaboration among multiple users over the Internet. Such applications include shared whiteboards, text and audio chat, shared slide shows, video, distance learning, and collaborative browsing of Web pages. These applications are characterized by the need to establish user groups and exchange information to ensure that all members of a particular group see consistent information (e.g. see the same whiteboard image, see and hear each others' text/audio, see slides/video in a synchronized manner, see the same Web page, etc). In effect, the consistent shared state updates are "pushed" to each client. At the same time, we need to provide freedom for each user to retrieve information by downloading arbitrary Web pages (a "pull" metaphor) and presenting that information to the collaborative application and group.

Typically, these collaborative systems run as standalone applications because of their need to exchange specific data types and their special "push" and "pull" requirements. Indeed, these applications provide their own custom user interface and do not interoperate with existing tools.

Systems like America Online's "Virtual Places" (VP) system have attempted to integrate collaboration into existing World-Wide Web browsers such as Netscape Navigator. When the collaboration requires the user to see a new Web page or slide, this system communicates directly with the Web Browser and instructs it to display the particular Web page. To perform this communication, VP relies on a Dynamic Data Exchange (DDE) interface exported by the particular Web Browser. DDE enables applications to exchange a pre-defined set of events, as supported by the particular endpoint applications.

The use of DDE restricts the VP functionality, however. First, DDE is only available on Microsoft Windows platforms. Furthermore, because the DDE interface is application-specific, the particular interface used by VP only works with the Netscape Navigator browser. In summary, the VP system does not provide a portable mechanism for integrating collaboration with the World-Wide Web.

Therefore, a need exists for a method and system for portably adding collaboration capabilities (both "push" and "pull" of information) to World-Wide Web content. In particular, a persistent connection is to be maintained to a shared state server for retrieving data via a "push" mechanism which maintains the freedom to access arbitrary Web content via the "pull" mechanism. Moreover, the collaboration should be provided in a portable manner, in the same way that all other Web content is portable both across hardware platforms and across browser implementations.

Summary of the Invention

It is an object of the present invention to provide a technique which alleviates the above drawbacks.

According to the present invention we provide a method of retrieving data from an information server, comprising the steps of: a requester initiating a request for data from an information server; sending said request to a Proxy; said Proxy forwarding said request to said information server; said Proxy receiving and storing data from said information server; inserting an Applet into said data; delivering said data and said Applet to said requester of said data; and executing said Applet.

Further according to the present invention we provide a system for retrieving data from an information server, comprising: means for a requester initiating a request for data from an information server; means for sending said request to a Proxy; means for said Proxy forwarding said request to said information server; means for said Proxy receiving and storing data from said information server; means for inserting an Applet into said data; means for delivering said data and said Applet to said requester of said data; and means for executing said Applet.

Also according to the present invention we provide a computer readable storage medium having computer readable program code means for retrieving data from an information server, the computer readable program code means comprising: computer readable program code means for a requester initiating a request for data from an information server; computer readable program code means for sending said request to a Proxy; computer readable program code means for said Proxy forwarding said request to said information server; computer readable program code means for said Proxy receiving and storing data from said information server; computer readable program code means for inserting an Ap-

plet into said data; computer readable program code means for delivering said data and said Applet to said requester of said data; and computer readable program code means for executing said Applet.

These and other features, aspects, and advantages of the present invention will become better understood with reference to the following description, appended claims, and accompanying drawings.

Brief Description of the Drawings

For a more complete understanding of the present invention and for further advantages thereof, reference is now made to the following Detailed Description taken in conjunction with the accompanying drawings, in which:

Figure 1 is a pictorial representation of a data processing system which may be utilized to implement a method and system of the present invention;

Figure 2 shows the data processing elements of a collaborative system that integrates with Web Browsers in a platform-independent manner;

Figure 3 is a flowchart depicting the steps required for delivering data from a Web Server to a Web Browser in a collaborative environment;

Figures 4A, 4B, 4C and 4D are flowcharts illustrating the actions performed by the inserted Applet when it executes in the Web Browser;

Figure 5 is a flowchart illustrating how a user joins a Tour; and

Figure 6 is a flowchart illustrating how a user is notified when a Tour leader moves to a new Web page.

Detailed Description of the Invention

Referring to Figure 1, there is depicted a graphical representation of a data processing system 8, which may be utilized to implement the present invention. As may be seen, data processing system 8 may include a plurality of networks, such as Local Area Networks (LAN) 10 and 32, each of which preferably includes a plurality of individual computers 12 and 30, respectively. Of course, those skilled in the art will appreciate that a plurality of Intelligent Work Stations (IWS) coupled to a host processor may be utilized for each such network. Each said network may also consist of a plurality of processors coupled via a communications medium, such as shared memory, shared storage, or an interconnection network. As is common in such data processing systems, each individual computer may be coupled to a storage device 14 and/or a printer/output device 16 and may be provided with a pointing device such as a mouse 17.

The data processing system 8 may also include multiple mainframe computers, such as mainframe computer 18, which may be preferably coupled to LAN 10 by means of communications link 22. The mainframe computer 18 may also be coupled to a storage device 20 which may serve as remote storage for LAN 10. Similarly, LAN 10 may be coupled via communications link 24 through, a sub-system control unit/communications controller 26 and communications link 34 to a gateway server 28. The gateway server 28 is preferably an IWS which serves to link LAN 32 to LAN 10.

With respect to LAN 32 and LAN 10, a plurality of documents or resource objects may be stored within storage device 20 and controlled by mainframe computer 18, as resource manager or library service for the resource objects thus stored. Of course, those skilled in the art will appreciate that mainframe computer 18 may be located a great geographic distance from LAN 10 and similarly, LAN 10 may be located a substantial distance from LAN 32. For example, LAN 32 may be located in California while LAN 10 may be located within North Carolina and mainframe computer 18 may be located in New York.

Software program code which employs the present invention is typically stored in the memory of a storage device 14 of a stand alone workstation or LAN server from which a developer may access the code for distribution purposes, the software program code may be embodied on any of a variety of known media for use with a data processing system such as a diskette or CD-ROM or may be distributed to users from a memory of one computer system over a network of some type to other computer systems for use by users of such other systems. Such techniques and methods for embodying software code on media and/or distributing software code are well-known and will not be further discussed herein.

Referring to Figure 2, the basic elements of a collaborative system that integrates with a Web Browser are shown. A Multi-User Shared State Server (MUSSS) 201 provides collaboration services on behalf of a set of users. This server is responsible for facilitating the exchange of information among collaborators. Although not shown, it is to be understood that the MUSSS may comprise a plurality of interconnected servers that together provide the services offered by a MUSSS.

A Proxy 202 runs on each client host. It includes Client Software 203 that communicates with the MUSSS 201. Although the Proxy resides on the client host in the preferred embodiment, alternative embodiments may place the Proxy on other hosts, such as on a well known server. Over this communication, a client provides information to the collaborative session and receives information from other participants in the collaboration. The Proxy 202 also contains Forwarding Software 204 that receives Web page requests from a local Web Browser 205 and forwards them to an appropriate

destination Web Server 206. The Forwarding Software 204 manipulates the retrieved data before delivering it to the Web Browser 205.

Finally, the Web Browser 205 displays the retrieved data from the Web Server 206 via Forwarding Software 204. The Browser 205 executes an Applet 207 that is embedded by the Forwarding Software 204 into the retrieved data. Applet 207 maintains a connection 208 to the Client Software 203 in the Proxy 202. Over connection 208, the Applet receives commands from the collaboration session and provides information that the Client Software 203 can disseminate to the collaboration session. Typically, the Applet 207 is written in the Java or JavaScript programming languages, which are standard mechanisms for delivering software to Web Browsers in a browser-independent and platform-independent manner.

Referring now to Figure 3, a flowchart depicts how Web data is retrieved by a Web Browser. At block 301, the Web Browser transmits a data request to the Forwarding Software embedded inside the Proxy. At block 302, the Forwarding Software initiates a connection to the appropriate Web Server. At block 303, the Web Server receives the request and transmits data to the Forwarding Software. At block 304, the Forwarding Software receives data from the Web Server and inserts an Applet into the data. At block 305, the Forwarding Software passes the retrieved data (with the inserted Applet) to the Browser. At block 306, the Browser receives the data, parses the data, displays the data, and executes the Applet. The procedure then terminates at 307. Although not shown, it is understood that blocks 303, 304, and 305 may occur concurrently as data is streamed from the Web Server to the Forwarding Software to the Web Browser. Although the Applet is inserted by the Forwarding Software in the preferred embodiment, alternative embodiments may have the Applet inserted by the Web Server or statically present within the page.

The Applet is responsible for initiating a connection to the Client Software (see Figure 2) so that the Web Browser may be linked with the collaboration even though arbitrary content has been loaded (pulled) into the Browser. When the Applet is inserted into the data retrieved from the Web Server, the Forwarding Software may provide parameter(s) that designate how that Applet should connect to the Client Software. In addition, when the Web page contains "frames" representing multiple documents that are simultaneously visible, multiple Applets may be active simultaneously. Consequently, the first Applet to run places itself into a global variable, so that multiple Applets do not communicate with the Client Software simultaneously from within a single Web Browser.

Once the Applet is executed, it sends an "addURL" message to inform the Client Software of which page is currently visible on the Browser window. When the Applet is terminated, it sends a "removeURL" message notifying the Client Software that the page is no longer visible. It may send "readData" and "writeDAta" messages to respectively retrieve and store persistent information from and to the Proxy. Finally, the Applet listens for commands from the Client Software such as, for example, a request to load a new Web page into the Browser.

Referring now to Figure 4A, a flowchart illustrates the actions performed by the Applet inserted by the Forwarding Software when it executes in the Web Browser. At decision block 401, the Applet checks whether or not the global variable is null. If the response to decision block 401 is yes, then in block 402, the Applet initializes the global variable to point to itself. In block 403, the Applet retrieves any connection parameters provided by the Forwarding Software. In block 404, the Applet initiates a connection to the Client Software. At block 405, the Applet sends an "addURL" message with the current page URL as a parameter over its connection to the Client Software. At this point, the Applet initiates three concurrent execution paths through blocks 406, 408, and 410. Referring to Fig. 4B, at decision block 406, the Applet determines whether a command to load a new URL is received from the Client Software. If the response to decision block 406 is yes, then at block 407, the Applet notifies the Browser to load a new page and then returns to decision block 406. Concurrently, at block 408 (Fig. 4C), the Applet determines whether or not it received a request from another Applet to add or remove a URL. If the response to decision block 408 is yes, then at block 409 the Applet sends an "addURL" or "removeURL" message over its connection to the Client Software and then returns to decision block 408. Concurrently, at decision block 410 (Fig. 4D), the Applet determines whether or not a stop message has been received from the Browser. If the response to decision block 410 is no, execution returns to decision block 410. If the response to decision block 410 is yes, then at block 411, the Applet sends a "removeURL" message to the Client Software with the current page URL as a parameter. At block 412, the Applet terminates the connection to the Client Software. At block 413, the Applet restores the global variable to null. It then terminates at block 414.

Continuing with Figure 4A, if the answer to decision block 401 is no, then at block 415, the Applet notifies the Applet represented by the global variable to add a URL. At this point, the Applet initiates two concurrent execution paths through blocks 416 and 417. At decision block 416, the Applet determines whether or not the global variable is null. If the response to decision block 416 is no, execution returns to block 416. If the response to decision block 416 is yes, execution continues to block 402. Concurrently, at block 417, the Applet determines whether or not a stop message has been received from the Browser. If the response to decision block 417 is no, execution returns to decision block 417. If the response to decision block 417 is yes, then at block 418, the Applet notifies the Applet represented by the global variable to remove this URL. The Applet then terminates at 414.

Within this collaborative system, the users may collaborate with each other according to whether they are currently viewing the same page. When the Client Software receives an "addURL" or "removeURL" notification, it forwards the information to the MUSSS. With information about the pages currently viewed by each user, the MUSSS is responsible for ensuring that collaborative data (including text and audio chat, whiteboard data, etc.) is exchanged among users who are seeing one or more documents in common on their respective Browsers or among groups of users as determined by the application.

Through this collaboration system, users may participate in a "Tour." In such a Tour, when a designated Leader "pulls" a new page into his Browser, the Browsers at one or more "followers" automatically load the same page. In this way, a group of users may concurrently "pull" content from the Web while the collaboration system "pushes" information to ensure that the pages seen by the users are consistent.

Referring now to Figure 5, the mechanis by which a user joins a Tour is illustrated. At block 501, the user selects a leader for the Tour. At block 502, the Client Software transmits a "follow" request including the leader's identity to the MUSSS. At decision block 503, the MUSSS determines whether or not the system is operating in a peer-to-peer mode. If the response to decision block 503 is no, then at block 504, the server stores an association between the leader and the follower. The procedure then terminates at 505. If the response to decision block 503 is yes, then at block 506, the server forwards the "follow" request to the leader who, at block 507, adds the follower to his own follower list. The procedure then terminates at 505. Although not depicted in Figure 5, it is to be understood that the path through blocks 502, 503, and 506 may be shortened if the Client Software can determine that the system is operating in a peer-to-peer mode; in this case, the Client Software can transmit the "follow" request directly to the leader, as represented by block 502-A.

Referring now to Figure 6, the mechanism by which followers are notified when a Tour leader moves to a new Web page is shown. In block 601, the Client Software receives an "addURL" message from the leader's Web Browser. At block 602, the Client Software forwards the "addURL" notification to the MUSSS. At decision block 603, it is determined whether or not the system is operating in a peer-to-peer mode. If the response to decision block 603 is no, then at block 604, the MUSSS sends "changePage" notifications to each of the leader's associated followers. At block 605, the Client Software at a follower receives a "changePage" notification from the MUSSS. At block 606, the Client Software forwards the "changePage" notification to the Applet running in the local Web Browser. The process then terminates at block 607. If the response to decision block 603 is yes, then at block 608, the Client Software at the leader's host sends "changePage" notifications to each of the followers in the local follower list. Control then continues to block 605, as described above.

Thus, the collaboration system can push information to all clients while simultaneously allowing them to pull arbitrary information from Web servers. Moreover, because control of the Browser is performed through an Applet written in Java, the collaboration system is independent of the particular machine or operating system environment and is independent of the particular Web Browser in use.

Claims

1. A method of retrieving data from an information server, comprising the steps of:

   a requester initiating a request for data from an information server;
   sending said request to a Proxy;
   said Proxy forwarding said request to said information server;
   said Proxy receiving and storing data from said information server; inserting an Applet into said data;
   delivering said data and said Applet to said requester of said data; and
   executing said Applet.

2. The method of Claim 1, wherein said step of executing said Applet further comprises the steps of:

   initiating a connection to Client Software; and
   terminating said connection to Client Software.

3. The method of Claim 2, further comprising the step of:

   sending an "addURL" notification over said connection to Client Software.

4. The method of Claim 2, further comprising the step of:

   sending a "removeURL" notification over said connection to Client Software.

5. The method of Claim 2, further comprising the step of:

   sending a "readData" request over said connection to Client Software.

6. The method of Claim 2, further comprising the step of:

   sending a "writeData" request over said connection to Client Software.

7. The method of Claim 2, further comprising the steps of:

receiving a "changePage" notification over said connection to Client Software; and
initiating the retrieval of new data in accordance with said "changePage" notification.

8. The method of claim 7, further comprising the steps of:

initiating a "follow" request for a target leader from said Client Software;
recording an identity of an originator of said "follow" request; and
initiating a "changePage" request to said originator as a result of actions taken by said target leader.

9. The method of Claim 2 wherein said Client Software communicates with a Multi-User Shared State Server.

10. The method of Claim 2, wherein said step of initiating a connection to Client Software includes determining an identity of a "global" Applet, and wherein said connection to said Client Software is a logical connection established by forwarding requests to said "global" Applet.

11. The method of Claim 10, further comprising the steps of:

monitoring for a presence of a "global" Applet; and
becoming said "global" Applet should there be no said "global" Applet present.

12. A system for retrieving data from an information server, comprising:

means for a requester initiating a request for data from an information server;
means for sending said request to a Proxy;
means for said Proxy forwarding said request to said information server;
means for said Proxy receiving and storing data from said information server;
means for inserting an Applet into said data;
means for delivering said data and said Applet to said requester of said data; and
means for executing said Applet.

13. The system of Claim 12, wherein said means for executing said Applet further comprises:

means for initiating a connection to Client Software; and
means for terminating said connection to Client Software.

14. The system of Claim 13, further comprising:

means for sending an "addURL" notification over said connection to Client Software.

15. The system of Claim 13, further comprising:

means for sending a "removeURL" notification over said connection to Client Software.

16. The system of Claim 13, further comprising:

means for sending a "readData" request over said connection to Client Software.

17. The system of Claim 13, further comprising:

means for sending a "writeData" request over said connection to Client Software.

18. The system of Claim 13, further comprising:

means for receiving a "changePage" notification over said connection to Client Software; and
means for initiating the retrieval of new data in accordance with said "changePage" notification.

19. The system of Claim 18, further comprising:

means for initiating a "follow" request for a target leader from said Client Software;
means for recording an identity of an originator of said "follow" request; and
means for initiating a "changePage" request to said originator as a result of actions taken by said target leader.

20. The system of Claim 13, wherein said Client Software communicates with a Multi-User Shared State Server.

21. The system of Claim 13, wherein said means for initiating a connection to Client Software includes means for determining an identity of a "global" Applet, and wherein said connection to said Client Software is a logical connection established by forwarding requests to said "global" Applet.

22. The system of Claim 21, further comprising:

means for monitoring for a presence of a "global" Applet; and
means for becoming said "global" Applet

should there be no said "global" Applet present.

23. A computer readable storage medium having computer readable program code means for retrieving data from an information server, the computer readable program code means comprising:

computer readable program code means for a requester initiating a request for data from an information server;
computer readable program code means for sending said request to a Proxy;
computer readable program code means for said Proxy forwarding said request to said information server;
computer readable program code means for said Proxy receiving and storing data from said information server;
computer readable program code means for inserting an Applet into said data;
computer readable program code means for delivering said data and said Applet to said requester of said data; and
computer readable program code means for executing said Applet.

24. The computer readable storage medium of Claim 23, wherein said computer readable program code means for executing said Applet further comprises:

computer readable program code means for initiating a connection to Client Software; and
computer readable program code means for terminating said connection to Client Software.

25. The computer readable storage medium of Claim 24, further comprising:

computer readable program code means for sending an "addURL" notification over said connection to Client Software.

26. The computer readable storage medium of Claim 24, further comprising:

computer readable program code means sending a "removeURL" notification over said connection to Client Software.

27. The computer readable storage medium of Claim 24, further comprising:

computer readable program code means sending a "readData" request over said connection to Client Software.

28. The computer readable storage medium of Claim 24, further comprising:

computer readable program code means sending a "writeData" request over said connection to Client Software.

29. The computer readable storage medium of Claim 24, further comprising:

computer readable program code means for receiving a "changePage" notification over said connection to Client Software; and
computer readable program code means for initiating the retrieval of new data in accordance with said "changePage" notification.

30. The computer readable storage medium of Claim 29, further comprising:

computer readable program code means for initiating a "follow" request for a target leader from said Client Software;
computer readable program code means for recording an identity of an originator of said "follow" request; and
computer readable program code means for initiating a "changePage" request to said originator as a result of actions taken by said target leader.

31. The computer readable storage medium of Claim 24, wherein said Client Software communicates with a Multi-User Shared State Server.

32. The computer readable storage medium of Claim 24, wherein said computer readable program code means for initiating a connection to Client Software includes computer readable program code means for determining an identity of a "global" Applet, and wherein said connection to said Client Software is a logical connection established by forwarding requests to said "global" Applet.

33. The computer readable storage medium of Claim 32, further comprising:

computer readable program code means for monitoring for a presence of a "global" Applet; and
computer readable program code means for becoming said "global" Applet should there be no said "global" Applet present.

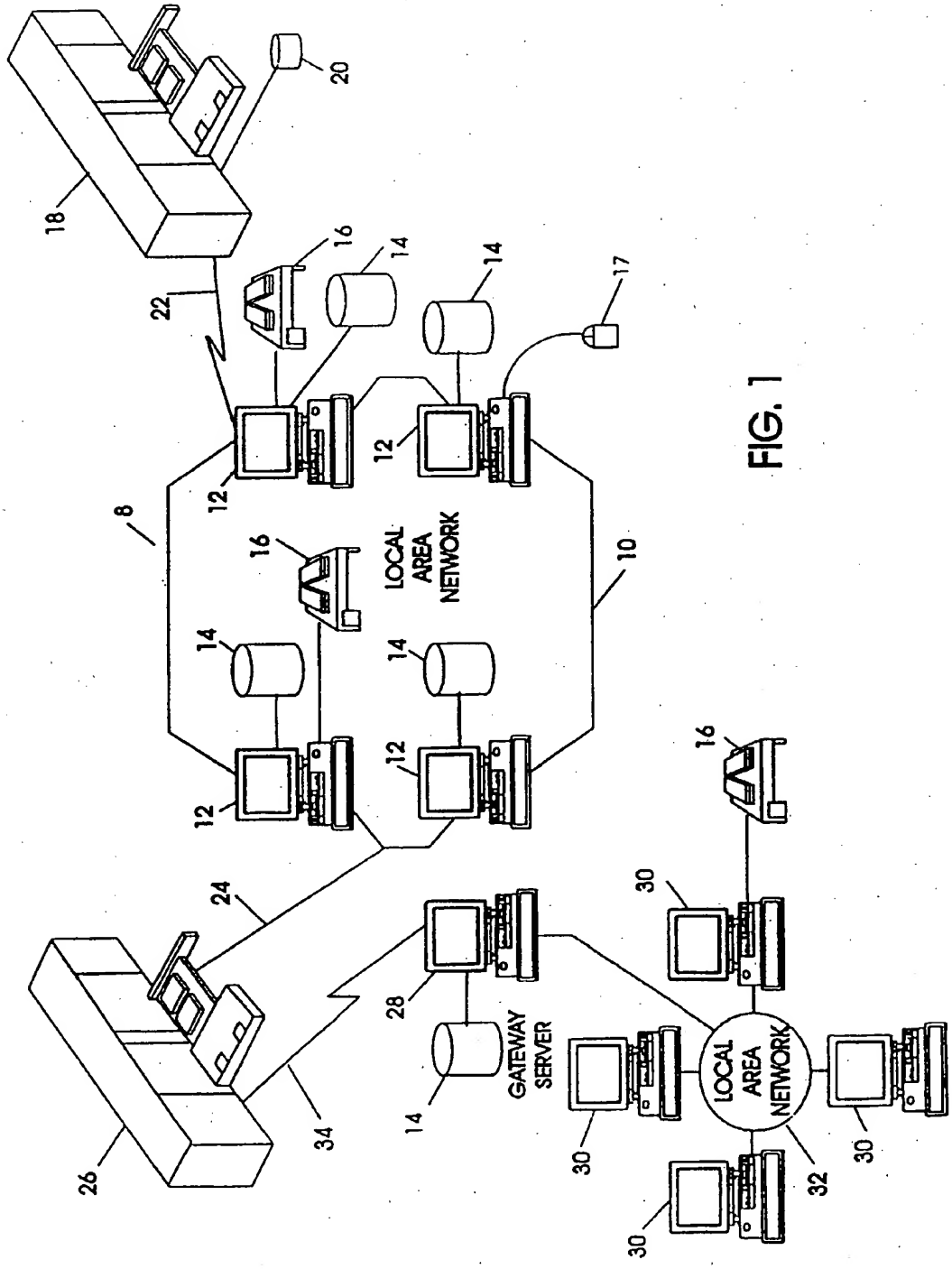


FIG. 1

FIG. 2

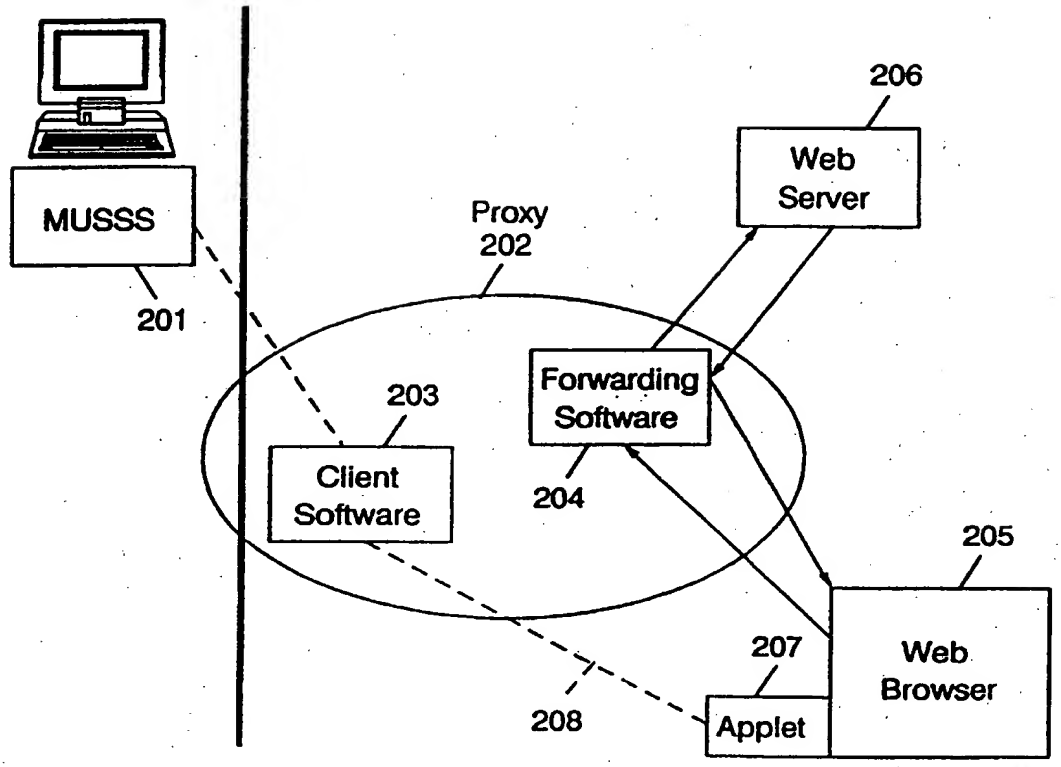

MUSSS
201
Proxy
202
206
Web
Server
203
Client
Software
Forwarding
Software
204
205
207
Web
Browser
208
Applet

FIG. 3

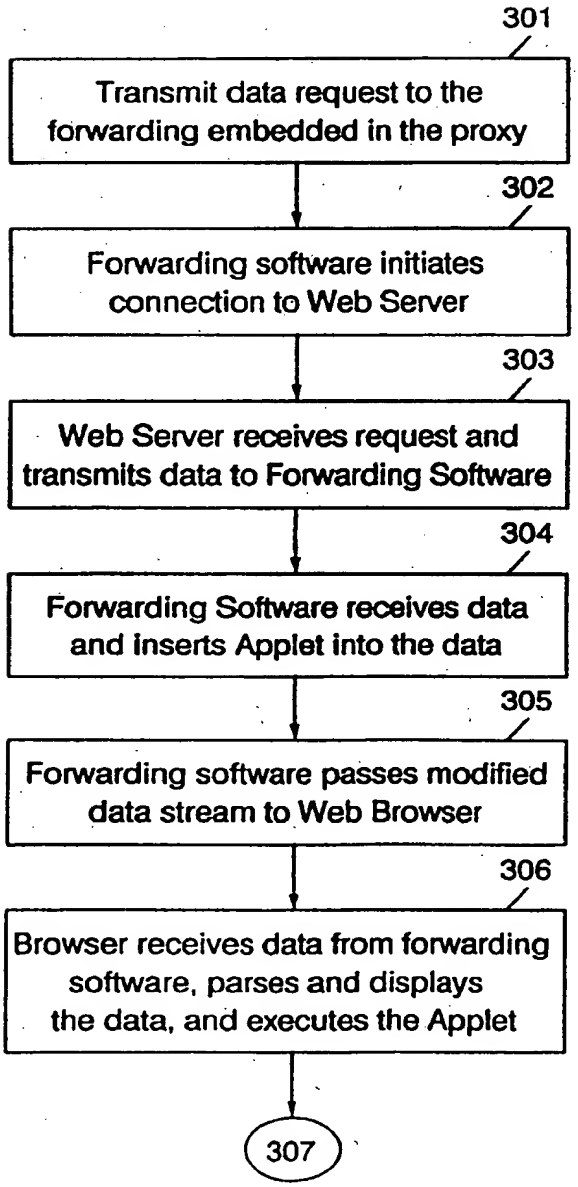

301
Transmit data request to the forwarding embedded in the proxy
302
Forwarding software initiates connection to Web Server
303
Web Server receives request and transmits data to Forwarding Software
304
Forwarding Software receives data and inserts Applet into the data
305
Forwarding software passes modified data stream to Web Browser
306
Browser receives data from forwarding software, parses and displays the data, and executes the Applet
307

FIG. 4A

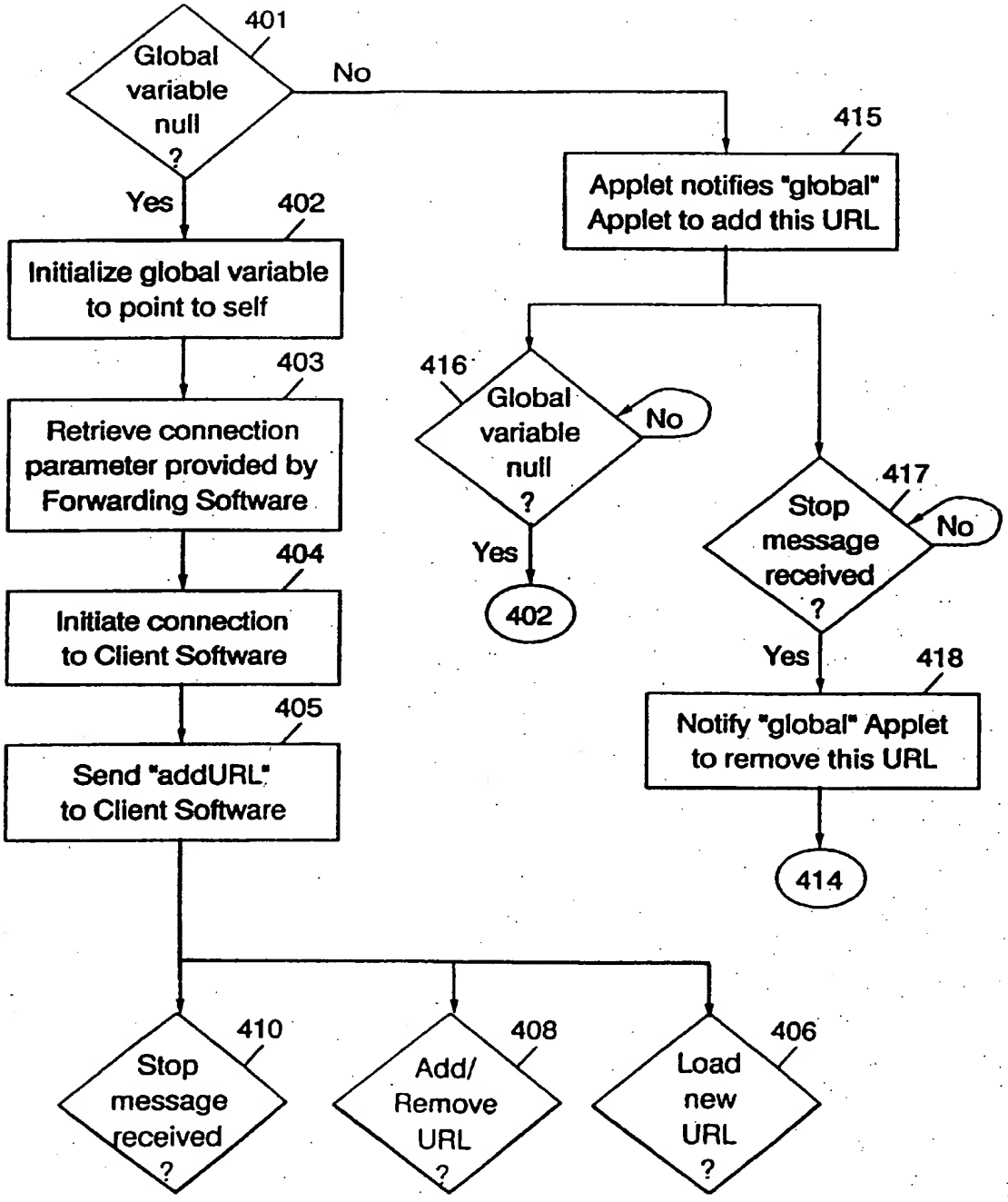

401
Global variable null ?
No
Yes
402
Initialize global variable to point to self
403
Retrieve connection parameter provided by Forwarding Software
404
Initiate connection to Client Software
405
Send "addURL" to Client Software
415
Applet notifies "global" Applet to add this URL
416
Global variable null ?
No
Yes
402
417
Stop message received ?
No
Yes
418
Notify "global" Applet to remove this URL
414
410
Stop message received ?
408
Add/ Remove URL ?
406
Load new URL ?

FIG. 4B

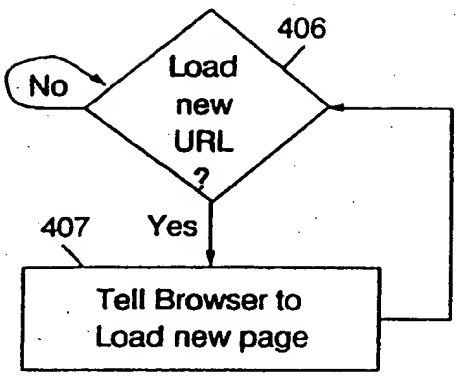


FIG. 4C

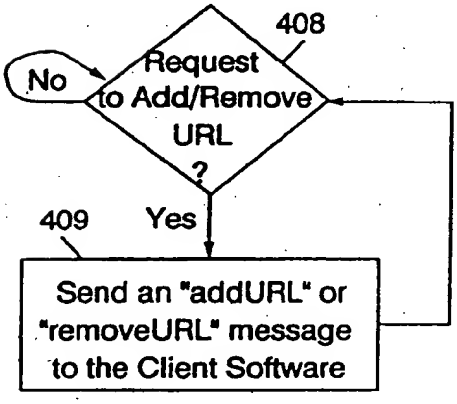


FIG. 4D

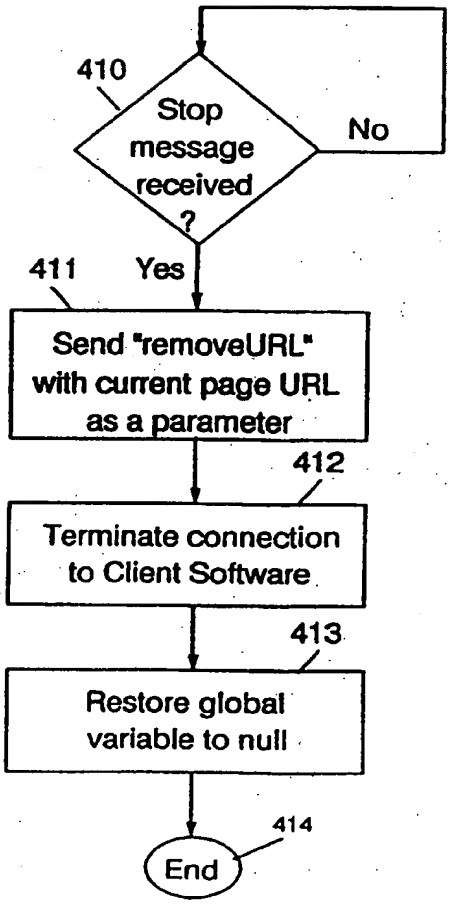
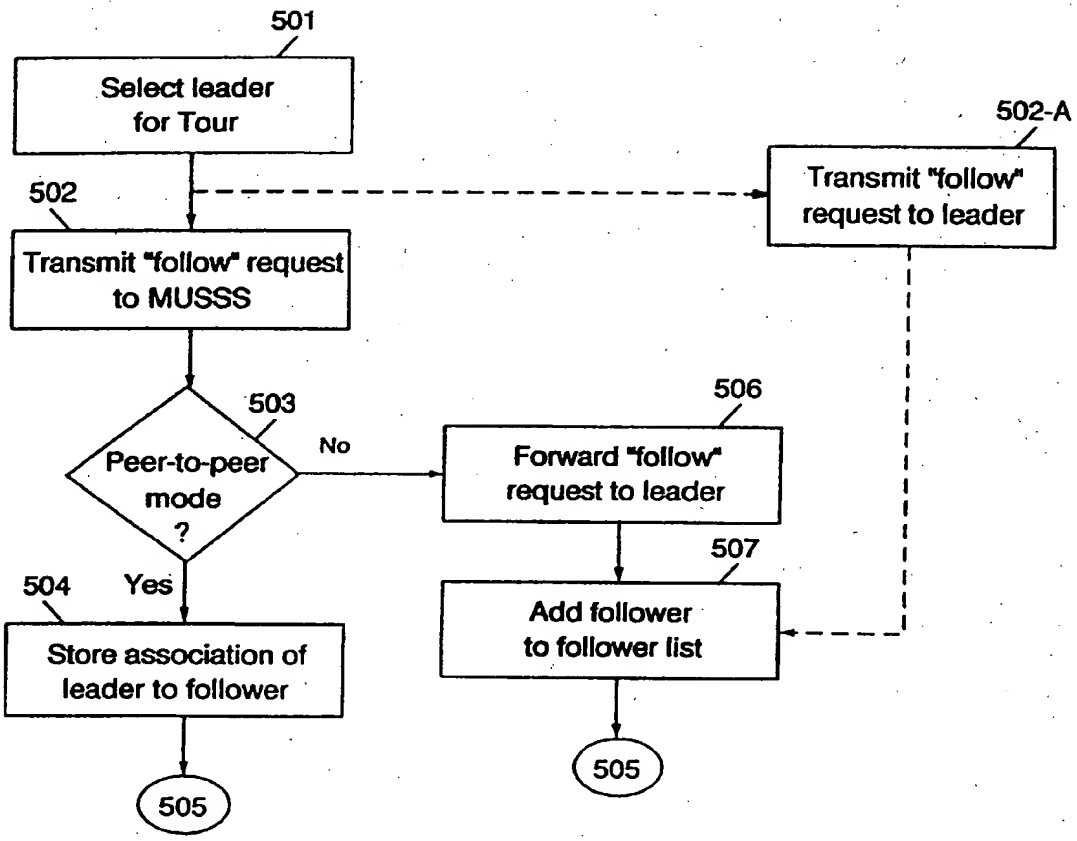


FIG. 5


501
Select leader for Tour
502
Transmit "follow" request to MUSSS
503
Peer-to-peer mode ?
No
Yes
504
Store association of leader to follower
505
502-A
Transmit "follow" request to leader
506
Forward "follow" request to leader
507
Add follower to follower list
505

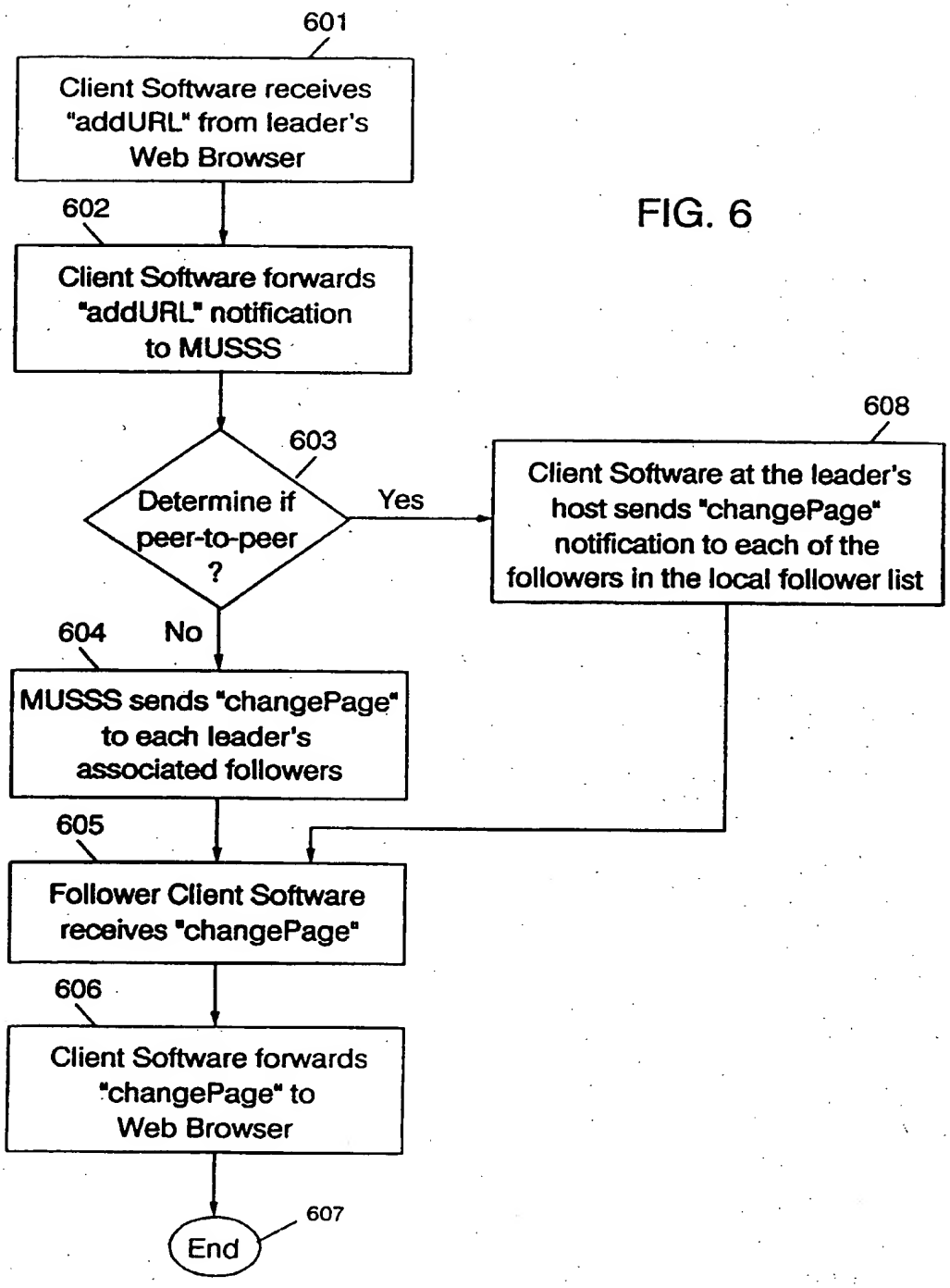

601
Client Software receives "addURL" from leader's Web Browser
602
Client Software forwards "addURL" notification to MUSSS
603
Determine if peer-to-peer ?
Yes
608
Client Software at the leader's host sends "changePage" notification to each of the followers in the local follower list
No
604
MUSSS sends "changePage" to each leader's associated followers
605
Follower Client Software receives "changePage"
606
Client Software forwards "changePage" to Web Browser
607
End

FIG. 6